Package 'trajectories'

August 28, 2014

Version 0.1-1

Date 2014-08-28

Title Classes and methods for trajectory data

Depends R (>= 2.14.0)

Imports methods, lattice, sp (>= 1.0-14), spacetime (>= 1.0-0), rgdal,rgeos

Suggests rgl, OpenStreetMap, RCurl, rjson, adehabitatLT

LazyData no

Description Classes and methods for trajectory data, with nested classes for individual trips, and collections for different entities. Methods include selection, generalization, aggregation, intersection, and plotting.

License GPL (>= 2)

URL http://github.com/edzer/trajectories

Collate Class-Tracks.R Tracks-methods.R stplot.R

Author Edzer Pebesma [cre, aut],Benedikt Klus [aut]

Maintainer Edzer Pebesma <edzer.pebesma@uni-muenster.de>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-28 12:56:00

R topics documented:

A3 *Trajectory*

Description

Trajectory, locally stored, from envirocar.org, see example below how it was imported

Usage

```
data(A3)
```

Examples

```
library(spacetime)
data(A3)
dim(A3)
## Not run:
importEnviroCar = function(trackID, url = "https://envirocar.org/api/stable/tracks/") {
require(RCurl)
require(rgdal)
require(rjson)
require(sp)
url = getURL(paste(url, trackID, sep = ""),
.opts = list(ssl.verifypeer = FALSE)) # .opts needed for Windows
# Read data into spatial object.
spdf = readOGR(dsn = url, layer = "OGRGeoJSON", verbose = FALSE)
# Convert time from factor to POSIXct.
time = as.POSIXct(spdf$time, format = "
# Convert phenomena from JSON to data frame.
phenomena = lapply(as.character(spdf$phenomenons), fromJSON)
values = lapply(phenomena, function(x) as.data.frame(lapply(x, function(y) y$value)))
# Get a list of all phenomena for which values exist.
names = vector()
for(i in values)
names = union(names, names(i))
# Make sure that each data frame has the same number of columns.
values = lapply(values, function(x) {
xNames = names(x)
# Get the symmetric difference.
diff = setdiff(union(names, xNames), intersect(names, xNames))
if(length(diff) > 0)
x[diff] = NA
x
})
# Bind values together.
data = do.call(rbind, values)
sp = SpatialPoints(coords = coordinates(spdf),
proj4string = CRS("+proj=longlat"))
stidf = STIDF(sp = sp, time = time, data = data)
Track(track = stidf)
```

```
}
A3 = importEnviroCar("528cf1a3e4b0a727145df093")

## End(Not run)
```

`generalize` *Generalize objects of class* `Track`, `Tracks` *and* `TracksCollection`

Description

Generalize objects of class `Track`, `Tracks` and `TracksCollection`.

Usage

```
## S4 method for signature 'Track'
generalize(t, FUN = mean, ...,  timeInterval, distance, n, tol, toPoints)
## S4 method for signature 'Tracks'
generalize(t, FUN = mean, ...)
## S4 method for signature 'TracksCollection'
generalize(t, FUN = mean, ...)
```

Arguments

`t`	An object of class `Track`, `Tracks` or `TracksCollection`.
`FUN`	The generalization method to be applied. Defaults to `mean` if none is passed.
`timeInterval`	(lower limit) time interval to split Track into segments
`distance`	(lower limit) distance to split Track into segments
`n`	number of points to form segments
`tol`	tolerance passed on to gSimplify, to generalize segments using the Douglas-Peucker algorithm.
`toPoints`	keep mid point rather than forming SpatialLines segments
`...`	Additional arguments passed to FUN

Value

An object of class `Track`, `Tracks` or `TracksCollection`.

stbox *obtain ranges of space and time coordinates*

Description

obtain ranges of space and time coordinates

Usage

```
stbox(obj)
```

Arguments

obj	object of a class deriving from Tracks or TracksCollection.

Value

stbox returns a data.frame, with three columns representing x-, y- and time-coordinates, and two rows containing min and max values. bbox gives a matrix with coordinate min/max values, compatible to bbox

Methods

stbox signature(x = "Tracks"): obtain st range from object

stbox signature(x = "TracksCollection"): obtain st range from object

stcube *Draw a space-time cube.*

Description

Draw a space-time cube.

Usage

```
## S4 method for signature 'Track'
stcube(x, xlab = "x", ylab = "y", zlab = "t", type = "l",
aspect, xlim, ylim, zlim, showMap = FALSE, mapType = "osm", ..., y, z)
## S4 method for signature 'Tracks'
stcube(x, xlab = "x", ylab = "y", zlab = "t", type = "l",
aspect, xlim, ylim, zlim, showMap = FALSE, mapType = "osm",
normalizeBy = "week", ..., y, z, col)
## S4 method for signature 'TracksCollection'
stcube(x, xlab = "x", ylab = "y", zlab = "t",
type = "l", aspect, xlim, ylim, zlim, showMap = FALSE, mapType = "osm",
normalizeBy = "week", ..., y, z, col)
```

Arguments

x	An object of class Track, Tracks or TracksCollection.
xlab, ylab, zlab, type, aspect, xlim, ylim, zlim	Arguments passed to plot3d() of package rgl.
showMap	Flag if a basemap is to be shown on the xy plane.
mapType	The tile server from which to get the map. Passed as type to openmap() of package OpenStreetMap.
normalizeBy	An abstract time period (either week or day) to be normalized by.
y, z, col	Ignored, but included in the method signature for implementation reasons.
...	Additional arguments passed to plot3d() of package rgl.

Value

A space-time cube.

Track-class *Classes "Track", "Tracks", and "TracksCollection"*

Description

Classes for representing sets of trajectory data, with attributes, for different IDs (persons, objects, etc)

Usage

```
  Track(track, df = NULL, fn = TrackStats)
  Tracks(tracks, tracksData = data.frame(row.names=names(tracks)),
   fn = TrackSummary)
  TracksCollection(tracksCollection, tracksCollectionData,
   fn = TracksSummary)
  TrackSummary(track)
  TracksSummary(tracksCollection)
  ## S4 method for signature 'Track'
x[i, j, ..., drop = TRUE]
  ## S4 method for signature 'TracksCollection'
x[i, j, ..., drop = TRUE]
  ## S4 method for signature 'Track,data.frame'
coerce(from, to)
  ## S4 method for signature 'Tracks,data.frame'
coerce(from, to)
  ## S4 method for signature 'TracksCollection,data.frame'
coerce(from, to)
```

Arguments

track	object of class STIDF-class, representing a single trip
df	optional data.frame with information between track points
tracks	named list with Track objects
tracksData	data.frame with summary data for each Track
tracksCollection	list, with Tracks objects
tracksCollectionData	data.frame, with summary data on tracksCollection
fn	function;
x	object of class Track etc
i	selection of spatial entities
j	selection of temporal entities (see syntax in package xts)
...	selection of attribute(s)
drop	logical
from	from
to	target class

Value

Functions Track, Tracks and TracksCollection are constructor functions that take the slots as arguments, check object validity, and compute summary statistics on the track and tracks sets.

Objects from the Class

Objects of class Track extend STIDF-class and contain single trips or tracks, objects of class Tracks contain multiple Track objects for a single ID (person, object or tracking device), objects of class TracksCollection contain multiple Tracks objects for different IDs.

Slots of class "Track"

sp: spatial locations of the track points, with length n

time: time stamps of the track points

endTime: end time stamps of the track points

data: data.frame with n rows, containing attributes of the track points

connections: data.frame, with n-1 rows, containing attributes between the track points such as distance and speed

Slots of class "Tracks"

tracks: list with Track objects, of length m

tracksData: data.frame with m rows, containing summary data for each Track object

Slots of class "TracksCollection"

`tracksCollection`: `list` `Tracks` objects, of length p

`tracksCollectionData`: `data.frame` with p rows, containing summary data for each `Tracks` object

Methods

[[`signature(obj = "Track")`: retrieves the attribute element

[[`signature(obj = "Tracks")`: retrieves the attribute element

[[`signature(obj = "TracksCollection")`: retrieves the attribute element

[[<- `signature(obj = "Track")`: sets or replaces the attribute element

[[<- `signature(obj = "Tracks")`: sets or replaces the attribute element

[[<- `signature(obj = "TracksCollection")`: sets or replaces the attribute element

$ `signature(obj = "Track")`: retrieves the attribute element

$ `signature(obj = "Tracks")`: retrieves the attribute element

$ `signature(obj = "TracksCollection")`: retrieves the attribute element

$<- `signature(obj = "Track")`: sets or replaces the attribute element

$<- `signature(obj = "Tracks")`: sets or replaces the attribute element

$<- `signature(obj = "TracksCollection")`: sets or replaces the attribute element

coerce Track,data.framecoerce to `data.frame`

coerce Tracks,data.framecoerce to `data.frame`

coerce TracksCollection,data.framecoerce to `data.frame`

plot `signature(x = "TracksCollection", y = "missing")`: plots sets of sets of tracks

stplot `signature(obj = "TracksCollection")`: plots sets of sets of tracks

Note

`segments` is a `data.frame` form in which track segments instead of track points form a record, with `x0`, `y0`, `x1` and `y1` the start and end coordinates

Author(s)

Edzer Pebesma, <edzer.pebesma@uni-muenster.de>

References

http://www.jstatsoft.org/v51/i07/

Examples

```
library(sp)
library(spacetime)
t0 = as.POSIXct(as.Date("2013-09-30",tz="CET"))
# person A, track 1:
x = c(7,6,5,5,4,3,3)
y = c(7,7,6,5,5,6,7)
n = length(x)
t = t0 + cumsum(runif(n) * 60)
require(rgdal)
crs = CRS("+proj=longlat")
#crs = CRS(as.character(NA))
stidf = STIDF(SpatialPoints(cbind(x,y),crs), t, data.frame(co2 = rnorm(n)))
A1 = Track(stidf)
# person A, track 2:
x = c(7,6,6,7,7)
y = c(6,5,4,4,3)
n = length(x)
t = max(t) + cumsum(runif(n) * 60)
stidf = STIDF(SpatialPoints(cbind(x,y),crs), t, data.frame(co2 = rnorm(n)))
A2 = Track(stidf)
# Tracks for person A:
A = Tracks(list(A1=A1,A2=A2))
# person B, track 1:
x = c(2,2,1,1,2,3)
y = c(5,4,3,2,2,3)
n = length(x)
t = max(t) + cumsum(runif(n) * 60)
stidf = STIDF(SpatialPoints(cbind(x,y),crs), t, data.frame(co2 = rnorm(n)))
B1 = Track(stidf)
# person B, track 2:
x = c(3,3,4,3,3,4)
y = c(5,4,3,2,1,1)
n = length(x)
t = max(t) + cumsum(runif(n) * 60)
stidf = STIDF(SpatialPoints(cbind(x,y),crs), t, data.frame(co2 = rnorm(n)))
B2 = Track(stidf)
# Tracks for person A:
B = Tracks(list(B1=B1,B2=B2))
Tr = TracksCollection(list(A=A,B=B))
stplot(Tr, scales = list(draw=TRUE))
stplot(Tr, attr = "direction", arrows=TRUE, lwd = 3, by = "direction")
stplot(Tr, attr = "direction", arrows=TRUE, lwd = 3, by = "IDs")
plot(Tr, col=2, axes=TRUE)
dim(Tr)
dim(Tr[2])
dim(Tr[2][1])
u = stack(Tr) # four IDs
dim(u)
dim(unstack(u, c(1,1,2,2))) # regroups to original
dim(unstack(u, c(1,1,2,3))) # regroups to three IDs
dim(unstack(u, c(1,2,2,1))) # regroups differently
```

```
as(Tr, "data.frame")[1:10,] # tracks separated by NA rows
as(Tr, "segments")[1:10,]   # track segments as records
Tr[["distance"]] = Tr[["distance"]] * 1000
Tr$distance = Tr$distance / 1000
Tr$distance
```

Index